Knowledge management
OWL and rules

Reasoning in OWL Ontologies

- Reasoners are used to infer information that isn't explicitly represented in an ontology.

- Terminological Reasoning
  - Consistency check - checks for inconsistent class definitions
  - Classification - determines the concepts that immediate subsume or are subsumed by a given concept
  - Taxonomy construction - computes all subclass relations (including those which are not explicitly stated but that are implied by the given definitions)

- Instance Reasoning
  - Instance checking - given a partial description of an individual (instance) and a class description, finds whether the class describes the instance
  - Individual retrieval - finds all instances that are described by a given concept

Why rules?

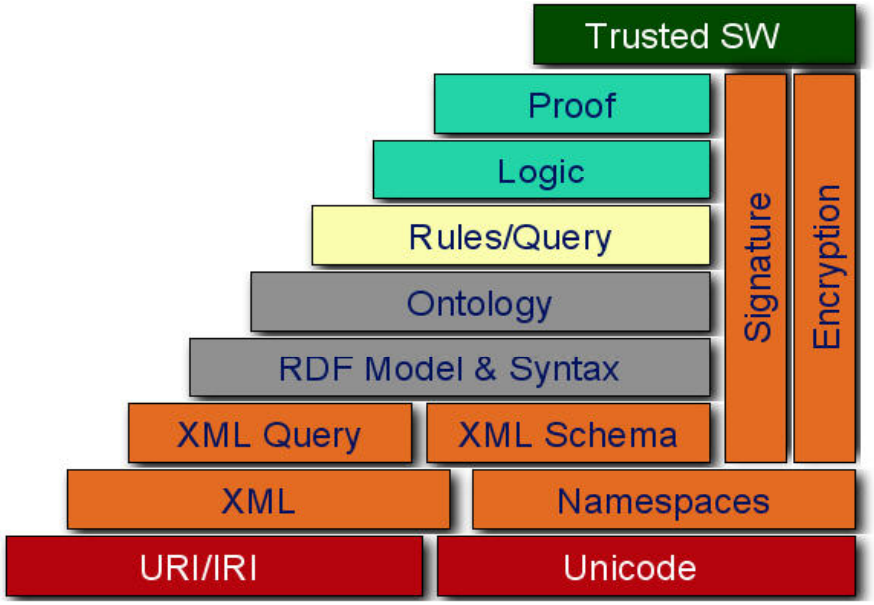
- OWL may not suffice for all applications
  - There are statements that cannot be expressed in OWL
  - Modeling constructs of OWL not always adequate or most desirable
  - First-order logic in general may be insufficient

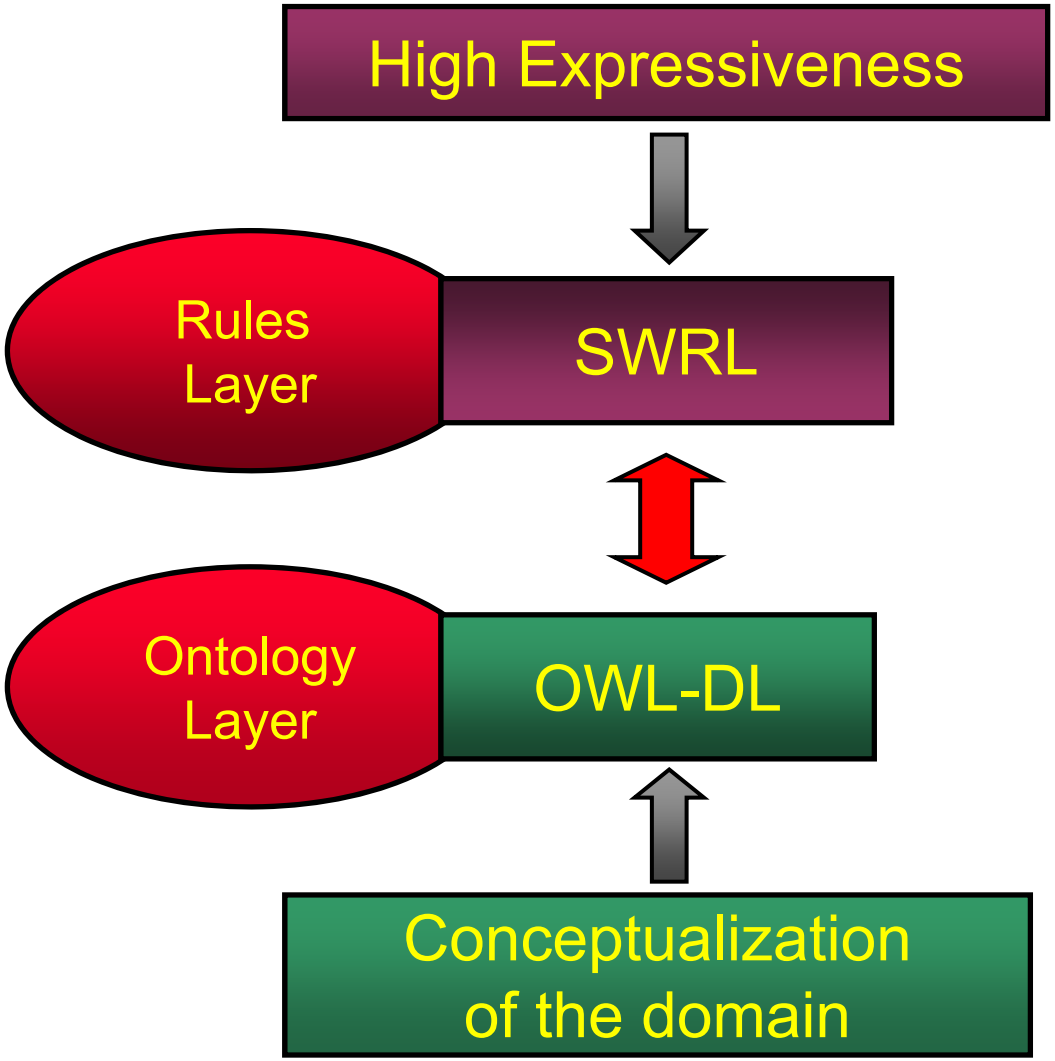
→ “Rules” as an alternative paradigm for knowledge modeling

We need both structure and rules

- **OWL's ontologies** are based on Description Logics (and thus in FOL)
  - ✓ The Web is an open environment.
  - ✓ Reusability / interoperability.
  - ✓ An ontology is a model easy to understand.
- Many **rule systems** based on logic programming
  - ✓ For the sake of decidability, ontology languages don't offer the expressiveness we want (e.g. constructor for composite properties?). Rules do it well.
  - ✓ Efficient reasoning support already exists.
  - ✓ Rules are well-known in practice.

A common approach





How to express Rules

- Rules can be expressed as first-order logic implications (Horn clauses)

$A1 \wedge A2 \ldots \wedge An \rightarrow H$ (“Body → Head”)

Example:

“$Student(x) \wedge happilyStudyingIn(x,y) \rightarrow HappyStudent(x)$”

- Constants, variables, function symbols can be used; but no negation

Where are the quantifiers?

- Quantifiers (forall, exists) are implicit
  - Variables only in **head** are existentially quantified
  - Variables in **body** are universally quantified
- Example:

$$isParent(X) \rightarrow hasChild(X,Y)$$

Is equivalent to

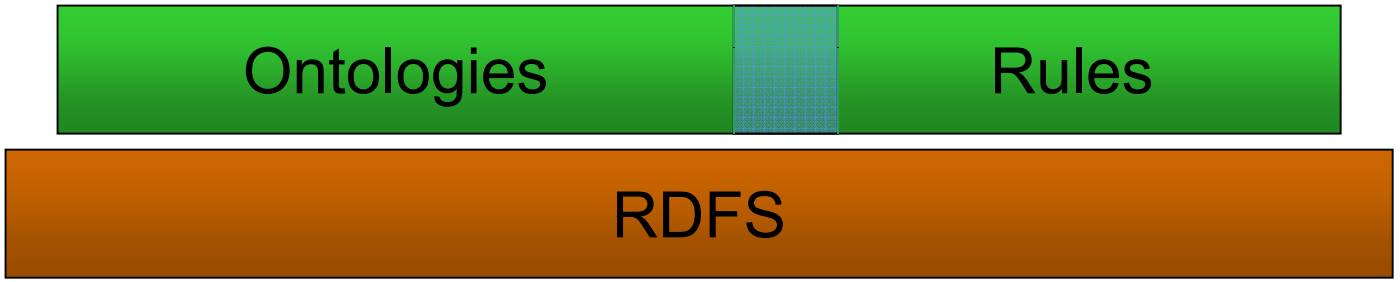
- Forall X : isParent(X) It exists Y : such as hasChild(X,Y)

Rules + Ontologies

- Still a challenging task!
- A number of different approaches exists: SWRL, DLP (Grosof), dl-programs (Eiter), DL-safe rules, Conceptual Logic Programs (CLP), AL-Log, DL+log
- Two main strategies:
  - Tight Semantic Integration (Homogeneous Approaches)
  - Strict Semantic Separation (Hybrid Approaches)

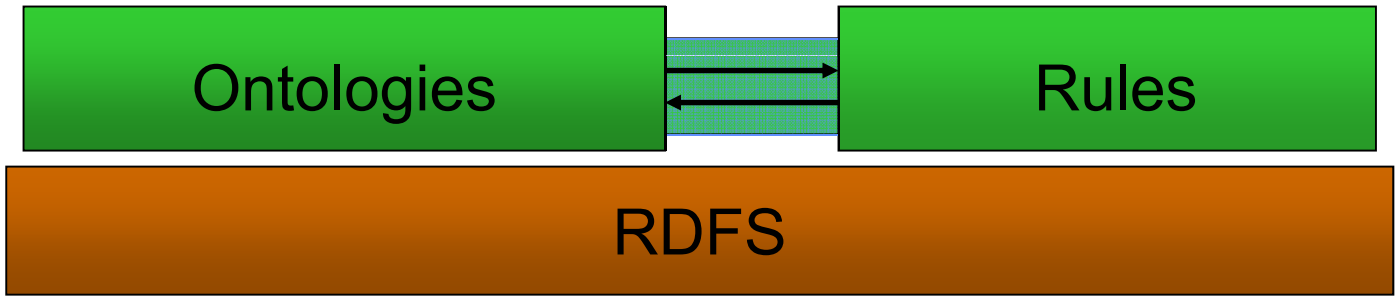
Homogeneous Approach

- Interaction with tight semantic integration.
- Both ontologies and rules are embedding in a common logical language.
- No distinction between rule predicates and ontology predicates.
- Rules may be used for defining classes and properties of the ontology.



Hybrid Approach

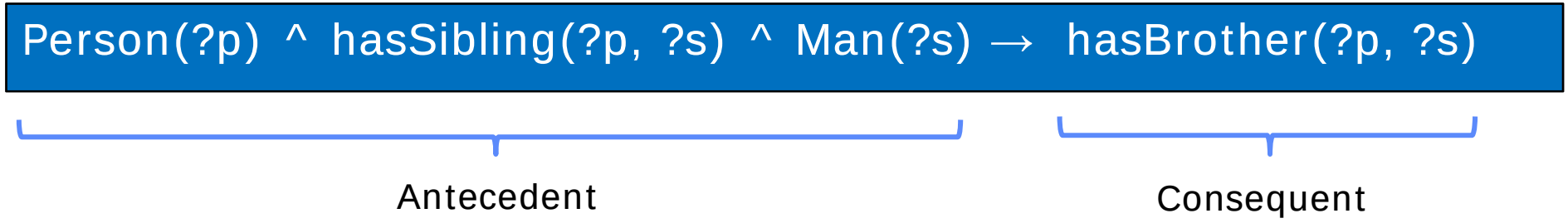
- Integration with strict semantic separation between the two layers.
- Ontology is used as a conceptualization of the domain.
- Rules cannot define classes and properties of the ontology, but some application-specific relations.
- Communication via a “safe interface”.



SWRL RULE LANGUAGE

SWRL

- SWRL – Semantic Web Rule Language
- W3C member submission: intended to be the rule language of the Semantic Web.
- Allows Horn-like rules to be combined with OWL-DL ontologies.
- Example SWRL Rule (**in Human Readable syntax**)
  - If A then B



SWRL Rule with Built-ins

- SWRL Built-ins provide functions for: comparison, math functions, string manipulations, etc.
- Swrl built-ins are identified using the http://www.w3.org/2003/11/swrlb namespace.
- Examples:

hasBrother(?x1,?x2) ^ hasAge(?x1,?age1) ^ hasAge(?x2,?age2) ^
swrlb:greaterThan(?age2,?age1)
→ hasOlderBrother(?x1,?x2)

hasBrother(?x1,?x2) ^ hasAge(?x1,?age1) ^ hasAge(?x2,?age2) ^
swrlb:subtract(10,?age2,?age1)
→ hasDecadeOlderBrother(?x1,?x2)

SWRL-Examples in XML concrete syntax

- Assert that the combination of the hasParent and hasBrother properties implies the hasUncle property in SWRL:

```
<ruleml:imp> <ruleml:_rlab ruleml:href="#example1"/>
   <ruleml:_body>
      <swrlx:individualPropertyAtom swrlx:property="hasParent">
         <ruleml:var>x1</ruleml:var>
         <ruleml:var>x2</ruleml:var>
      </swrlx:individualPropertyAtom>
      <swrlx:individualPropertyAtom swrlx:property="hasBrother">
         <ruleml:var>x2</ruleml:var>
         <ruleml:var>x3</ruleml:var>
      </swrlx:individualPropertyAtom>
   </ruleml:_body>
   <ruleml:_head>
      <swrlx:individualPropertyAtom swrlx:property="hasUncle">
         <ruleml:var>x1</ruleml:var>
         <ruleml:var>x3</ruleml:var>
      </swrlx:individualPropertyAtom>
   </ruleml:_head>
</ruleml:imp>
```

EXERCISES:

Express the SWRL code as a human-readable First Order Logic rule (Horn clauses).

Exercise 1

```
<ruleml:imp><ruleml:_rlab ruleml:href="#example2"/>
   <ruleml:_body>
      <swrlx:individualPropertyAtom swrlx:property="hasParent">
         <ruleml:var>x1</ruleml:var>
         <ruleml:var>x2</ruleml:var>
      </swrlx:individualPropertyAtom>
      <swrlx:individualPropertyAtom swrlx:property="hasSibling">
         <ruleml:var>x2</ruleml:var>
         <ruleml:var>x3</ruleml:var>
      </swrlx:individualPropertyAtom>
      <swrlx:individualPropertyAtom swrlx:property="hasSex">
         <ruleml:var>x3</ruleml:var>
         <owlx:Individual owlx:name="#male" />
      </swrlx:individualPropertyAtom>
   </ruleml:_body>
   <ruleml:_head>
      <swrlx:individualPropertyAtom swrlx:property="hasUncle">
         <ruleml:var>x1</ruleml:var>
         <ruleml:var>x3</ruleml:var>
      </swrlx:individualPropertyAtom>
   </ruleml:_head>
</ruleml:imp>
```

Exercise 2

```
<ruleml:imp>
 <ruleml:_rlab ruleml:href="#employsAndWorksForRule"/>
  <ruleml:_body>
    <swrlx:individualPropertyAtom swrlx:property="&org;employs">
      <ruleml:var>X</ruleml:var>
      <ruleml:var>Y</ruleml:var>
    </swrlx:individualPropertyAtom>
  </ruleml:_body>
  <ruleml:_head>
     <swrlx:individualPropertyAtom swrlx:property="&org;worksFor">
       <ruleml:var>Y</ruleml:var>
       <ruleml:var>X</ruleml:var>
     </swrlx:individualPropertyAtom>
  </ruleml:_head>
</ruleml:imp>
```

Exercise 3

```
<ruleml:imp> <ruleml:_rlab ruleml:href="#example1"/>
  <ruleml:_body>
    <swrlx:individualPropertyAtom swrlx:property="hasSister">
      <ruleml:var>x1</ruleml:var>
      <ruleml:var>x2</ruleml:var>
    </swrlx:individualPropertyAtom>
    <swrlx:individualPropertyAtom swrlx:property="hasBrother">
      <ruleml:var>x2</ruleml:var>
      <ruleml:var>x1</ruleml:var>
    </swrlx:individualPropertyAtom>
    <swrlx:individualPropertyAtom swrlx:property="hasMother">
      <ruleml:var>x1</ruleml:var>
      <ruleml:var>x3</ruleml:var>
    </swrlx:individualPropertyAtom>
  </ruleml:_body>
  <ruleml:_head>
    <swrlx:individualPropertyAtom swrlx:property="isMale">
      <ruleml:var>x1</ruleml:var>
    </swrlx:individualPropertyAtom>
    <swrlx:individualPropertyAtom swrlx:property="isFemale">
      <ruleml:var>x2</ruleml:var>
    </swrlx:individualPropertyAtom>
    <swrlx:individualPropertyAtom swrlx:property="hasMother">
      <ruleml:var>x2</ruleml:var>
      <ruleml:var>x3</ruleml:var>
    </swrlx:individualPropertyAtom>
  </ruleml:_head>
</ruleml:imp>
```